Prediction and compensation of contour error of CNC systems based on LSTM neural-network

Article (Accepted Version)

Prediction and Compensation of Contour Error of CNC Systems Based on LSTM Neural-Network

Jiangang Li, *Senior Member, IEEE,* Changgui Qi, Yanan Li*, *Senior Member, IEEE*, and Zenghao Wu

***Abstract*—This paper proposes a contour error estimation and compensation method for computer numerical control (CNC) systems based on the long short-term memory neural network (LSTM-NN). This is achieved by performing modeling of each axis to predict the tracking error, calculating the actual trajectory, estimating the contour error, and modifying the reference trajectory. First, linear feature selection based on a simplified single-axis model and nonlinear feature selection based on a circular test are performed to achieve tracking error prediction. Then, a spline-approximation-based contour error estimation method is proposed to estimate the contour error between the reference trajectory and the predicted trajectory. Finally, contour error compensation is performed on the reference trajectory before it is run on CNC systems. The proposed method is validated through experiments on a three-axis CNC system.**

***Index Terms*—CNC systems, Contour error, Neural network**

I. Introduction

The objective of high-speed and high-precision computer numerical control (CNC) machining is usually achieved through two main approaches, i.e. trajectory tracking control [1], [2] and contouring control [3]. Tracking control aims to improve the trajectory tracking performance of CNC systems and has been extensively studied in the early literature [4], [5]. In comparison, contouring control aims to improve the contouring performance of CNC systems. Since the machining performance is usually evaluated based on the contour error, contouring control has been mainly focused on in the latest literature of CNC [6], [7]. The contour error is defined as the shortest distance between the reference trajectory and the actual trajectory [8], [9]. Many research works indicate that the contour error is closely related to the mismatch between different axes [10], [11], [12], so contouring control should be designed to compensate for this mismatch. Therefore, two aspects need to be considered while performing contouring control: contour error estimation (CEE) and contour error control (CEC).

On the one hand, many methods have been developed to accurately estimate the contour error over the past years. In [13], an approach was proposed to find the closest reference sampling position to the actual sampling position by circle-approximation, and the distance between them is defined as the contour error at this sampling instant. In [14], the tangent vector on the reference trajectory was used to approximate the reference trajectory so that the distance from the actual sampling position to the tangent vector could be used to estimate the contour error. In [15], Hermite-spline was used to approximate actual sampling positions, and the distance from the reference sampling position to the spline was defined as the contour error. These methods perform well when the sampling frequency is high, but suffer from large estimation errors at a low sampling frequency.

On the other hand, most of contouring control methods are based on the cross-coupling control (CCC) or iterative learning control (ILC). The main idea of CCC is that individual axis controller considers not only the error of a single axis but also the errors of other axes, ensuring coordination between these axes [8], [9], [14], [16]. CCC has limited performance for high-speed, large-curvature trajectories due to the time-consuming contour error estimation algorithm and low robustness. ILC has been studied for contouring control in many research works [15], [17], which works well in repetitive tasks with iterative modification of the reference trajectory or the control input according to the contour error. In general, CCC and ILC belong to post-compensation algorithms as the error compensation is performed after the contour error has been generated. In this paper, we study pre-compensation for the contour error, which relies on the contour error prediction to be discussed in the following.

Artificial neural networks (ANNs) are widely used in various learning systems and can approximate any continuous function in a compact set. In [18], to achieve a quadcopter's trajectory tracking, deep neural networks (DNNs) were used to fit the inverse model of the dynamic quadcopter system. The desired trajectory was used as the input of the fitted inverse model and the reference trajectory was obtained as the model output. In [19], an approach combining ANNs and ILC was proposed to improve tracking performance of multi-axis industrial robots. For a given desired trajectory, a high-fidelity dynamic simulator was used to iteratively refine the external instructions to compensate for the robot's inner-loop dynamics. The desired trajectories and the corresponding refined input trajectories were used to train a multi-layer neural network to approximate the nonlinear inner-loop inverse dynamics model. In [20], a nonlinear autoregressive network with exogenous inputs (NARX) was used for real-time contour error estimation and compensation of CNC systems. The well-trained single-axis NARX model was used to predict the output position at the next sampling instant and the predicted output position was used to estimate the contour error. According to the estimated contour error, the error compensation was performed on the reference position to reduce the contour error in the

This research is supported by the National Key R&D Program of China under Grant 2019YFB1703700.

J. Li, C. Qi and Z. Wu are with the School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen, China 518055.

*Correspondence: Y. Li is with the Department of Engineering and Design, University of Sussex, Brighton BN1 9RH, UK. Email:yl557@sussex.ac.uk

next sampling instant. In [21], a feedforward input generation scheme based on neural network prediction was proposed for six-degree-of-freedom industrial robots, which was shown to have superior performance in position tracking and residual vibration suppression. In [22], a deep gated recurrent unit (GRU) neural network was adopted to predict the contour error of the multi-axis motion system, and the predicted contour error was used as pre-compensation. The neural network feature selection in these studies mainly adopts a heuristic approach and only considers linear features, such as velocity or acceleration, leading to low prediction accuracy.

Based on the above discussions, this paper will introduce a contour error prediction and compensation method, which includes LSTM-NN-based tracking error prediction and spline-interpolation-based contour error estimation. First of all, modelling of the CNC system is conducted to select appropriate linear features for the LSTM-NN while circular test is performed to select appropriate nonlinear features. Then, LSTM-NN training is performed to obtain the predicted tracking error of each axis. A contour error estimation and compensation algorithm is developed and performed on the reference trajectory. Two comparative experiments are carried out to compare several contour error estimation methods and different features of LSTM-NN, respectively.

The main contributions of this paper lie in two aspects. On the one hand, we introduce a new contour error estimation method based on spline approximation, which will be shown to perform better than other methods in [13], [14], [15] for low-sampling-frequency trajectories. On the other hand, we develop a neural network that includes nonlinear features in its inputs, which will be shown to improve the trajectory prediction accuracy and eventually improve the contour error compensation performance.

The rest of this paper is organized as follows. Single-axis model analysis is performed in Section II and LSTM-NN off-line training is explained in Section III. Section IV introduces the contour error estimation and compensation algorithm. Section V presents results of experiments. At last, we draw the conclusions of this paper.

II. System modelling

First, we need to analyze the main factors affecting the tracking error by building a simplified model of a single-axis servo system for the linear feature selection. As shown in Fig. 1, a single-axis servo system can be generally divided into three parts: servo drive, servo motor and mechanical transmission, for which the modeling will be respectively introduced.

A. Mechanical transmission

The dynamics equation of the mechanical transmission part is

$$J_L \frac{d^2\theta_S(t)}{dt^2} + B_L \frac{d\theta_S(t)}{dt} + T_S(t) = T_L(t) = K_L[\theta_M(t) - \theta_S(t)] \tag{1}$$

where J_L is the total moment of inertia, B_L is the total damping factor, K_L is the total stiffness, θ_S is the output angle,

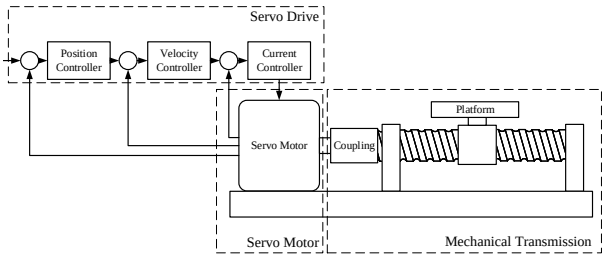


Fig. 1: A single-axis servo system

θ_M is the desired angle, T_S is the disturbance and T_L is the total torque.

For analysis convenience, we consider $T_S = 0$ and convert rotational motion into linear motion, and then perform a Laplace transformation to obtain the transfer function of the mechanical transmission part:

$$G_s(s) = \frac{x(s)}{\theta_M(s)} = \frac{l}{2\pi} \frac{k_L}{J_L s^2 + B_L s + k_L} \tag{2}$$

where $x(s) = i_S \theta_S(s)$ with $i_S = l/2\pi$ and l lead of screw.

B. Servo motor

Servo motors are usually permanent magnet synchronous motors. With the $d-q$ coordinate system, we use the following four equations to establish the mathematical model of permanent magnet synchronous motors:

1) voltage equation:

$$\begin{cases} U_q = Ri_q + \frac{d\psi_q}{dt} + w\psi_d \\ U_d = Ri_d + \frac{d\psi_d}{dt} - w\psi_q \end{cases} \tag{3}$$

where U_d and U_q are respectively the voltages of axes d and q, R is the stator resistance, ψ_d and ψ_q are respectively the magnetic flux of axes d and q, i_d and i_q are respectively the currents of axes d and q, and w is angular velocity of the rotor.

2) magnetic chain equation:

$$\begin{cases} \psi_q = L_q i_q \\ \psi_d = L_d i_d + \psi_f \end{cases} \tag{4}$$

where L_d and L_q are respectively the equivalent inductances of axes d and q, and ψ_f is the magnetic field of the permanent magnet across the magnetic chain of the stator winding.

3) electromagnetic torque equation:

$$T_e = K_c i_q \tag{5}$$

where $K_c = n_p \psi_f$ with T_e as the electromagnetic torque and n_p the polar logarithm of the stator.

4) motion equation:

$$T_e = T_L + J\frac{dw_r}{dt} + Dw_r \tag{6}$$

where T_L is the motor's load torque, J is the rotational inertia, D is the motor's damping coefficient and w_r is the motor's angular velocity $w_r = \frac{w}{n_p}$.

Based on the above four equations and considering $L_q = L_d = L$, $D = 0$, and $i_d = 0$, we can obtain a simplified model of a permanent magnet synchronous motor as below:

$$\begin{bmatrix} \dot{i}_q \\ \dot{w}_r \end{bmatrix} = \begin{bmatrix} -R/L & -n_p w_r/L \\ n_p \psi_f/J & 0 \end{bmatrix} \begin{bmatrix} i_q \\ w_r \end{bmatrix} + \begin{bmatrix} U_q/L \\ -T_L/J \end{bmatrix} \tag{7}$$

C. Servo drive

The servo drive includes an inverter, a low-pass filter, a current loop controller, a velocity loop controller and a position loop controller, for which the modelling will be carried out respectively.

The inverter can be modeled as G_{PWM}=$\frac{k_{pwm}}{\tau_{pwm}s+1}$ where k_{pwm} and τ_{pwm} are gain and time constant of the inverter respectively [23], while the low-pass filter $\frac{1}{1+\tau_c s}$ where τ_c is a time constant. By performing a three-loop rectification, we can obtain the following three controllers:

1) current loop controller $K_{CP}(\frac{T_{CI}s+1}{T_{CI}s})$ where K_{CP} and T_{CI} are the proportional and integral gains of the current loop controller respectively;
2) velocity loop controller $K_{VP}\frac{T_{VI}s+1}{T_{VI}s}$ where K_{VP} and T_{VI} are the proportional and integral gains of the velocity loop controller respectively;
3) position loop controller K_{PP} where K_{PP} is the proportional gain of the position loop controller.

By combining these models, we obtain a 6th order transfer function in Fig. 2, where $K = \frac{K_{VP}K_{CP}30n_p}{T_{VI}J\pi}$, $\tau = T_{VI}$, T=$\frac{T_{CI}}{K_{CP}}$, and v is the control input.

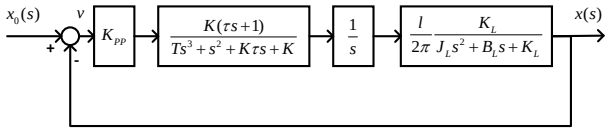


Fig. 2: Servo system's simplified block diagram

Moreover, we obtain the tracking error transfer function as

$$\phi_e(s) = \frac{s^6 + b_5 s^5 + b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s}{s^6 + a_5 s^5 + a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} \quad (8)$$

where

$$a_5 = b_5 = \frac{J_L+TB_L}{TJ_L},\ a_4 = b_4 = \frac{B_L+TK_L+J_LK\tau}{TJ_L},$$
$$a_3 = b_3 = \frac{KJ_L+K_L+B_LK\tau}{TJ_L}, a_2 = b_2 = \frac{BK_L+K_LK\tau}{TJ_L},$$
$$a_1 = \frac{2\pi KK_L+KK_{PP}K_Ll\tau}{2\pi TJ_L}, a_0 = \frac{KK_{PP}K_Ll}{2\pi TJ_L}, b_1 = \frac{KK_L}{TJ_L}.$$

III. Construction and Training of LSTM-NN

The modelling in the previous section lays the foundation of the analysis of the single-axis tracking error, based on which we will select neural network for the prediction and determine features for training of the neural network.

A. Feature selection

1) Linear feature selection based on simplified mathematical model: By performing inverse Laplace transformation on Eq. (8), we obtain the complete model of the single-axis system:

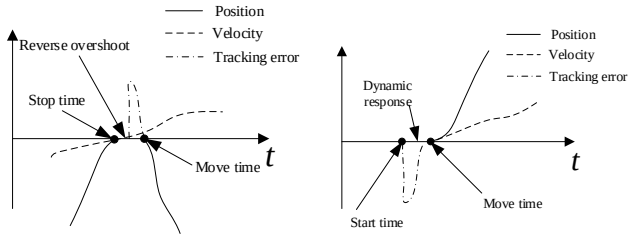
$$e(t) = \frac{1}{a_0}[\sum_{i=1}^{6} b_i x^{(i)}(t) - \sum_{j=1}^{6} a_j e^{(j)}(t)] \quad (9)$$

where $x^{(i)}(t)$ denotes the ith-order derivative of x with reference to time t and $e^{(j)}(t)$ the jth-order derivative of e. By converting Eq. (9) from continuous time domain to discrete time domain with sampling instant k, we obtain

$$e(k) = \sum_{i=-6}^{6} B_i \dot{x}(k+i) + \sum_{j=-6}^{6} A_j \dot{e}(k+j) \quad (10)$$

where B_i and A_j are modulus calculated from Eq. (9). This equation indicates that the tracking error e at sampling instant k is a result of $\dot{x}(k+i), i = -6, -5 \ldots 6$ and $\dot{e}(k+j), j = -6, -5 \ldots 6$. Therefore, $\dot{x}(k+i), i = -6, -5 \ldots 6$ are used as the linear features of the neural network while $\dot{e}(k+j), j = -6, -5 \ldots 6$ will be fitted by long-term memory and short-term memory of LSTM-NN.

2) Nonlinear feature selection based on circular test: A large overshoot will lead to tracking error due to backlash or dynamic response of a motor when it starts moving, which can be identified in a circular motion [24], [25] (see Fig. 3). Therefore, we perform a circular test to identify the backlash and dynamic response, which will be considered as nonlinear features of the LSTM-NN.



(a) overshoot when motor moves in reverse direction (b) overshoot when motor starts moving

Fig. 3: Tracking error caused by overshoot

As shown in Fig. 3(a), the reference velocity passes zero when the actual position is in reverse direction. According to this phenomenon, we use the reference velocity to mark the backlash interval:

$$\begin{cases} ch_1(k:k+n_1) = 1, \text{ if } v_{k+1} \times v_k < 0 \text{ and } v_{k+1} > 0; \\ ch_1(k:k+n_1) = -1, \text{ if } v_{k+1} \times v_k < 0 \text{ and } v_{k+1} < 0; \\ ch_1(k) = 0, \text{ otherwise.} \end{cases} \quad (11)$$

where k represents the sampling instant, v_k is the reference velocity at instant k, $k:k+n_1$ represents the interval from k to $k+n_1$, and n_1 is the window size of backlash.

The overshoot due to the motor's dynamic response is illustrated in Fig. 3(b), showing that the reference velocity starts from zero when the motor starts moving. According to this phenomenon, we use the reference velocity again to mark

the motor-starting interval:

$$\begin{cases} ch_2(k:k+n_2)=1, \text{ if } \begin{cases} v_{k+1}\times v_k=0,\ v_{k+1}>0 \\ \frac{1}{n_2}\sum_{i=0}^{n_2}|a_i| <= \frac{1}{4}|a_{max}| \end{cases} \\ ch_2(k:k+n_2)=-1, \text{ if } \begin{cases} v_{k+1}\times v_k=0,\ v_{k+1}<0 \\ \frac{1}{n_2}\sum_{i=0}^{n_2}|a_i| <= \frac{1}{4}|a_{max}| \end{cases} \\ ch_2(k:k+n_2)=2, \text{ if } \begin{cases} v_{k+1}\times v_k=0,\ v_{k+1}>0 \\ \frac{1}{n_2}\sum_{i=0}^{n_2}|a_i| > \frac{1}{4}|a_{max}| \end{cases} \\ ch_2(k:k+n_2)=-2, \text{ if } \begin{cases} v_{k+1}\times v_k=0,\ v_{k+1}<0 \\ \frac{1}{n_2}\sum_{i=0}^{n_2}|a_i| > \frac{1}{4}|a_{max}| \end{cases} \\ ch_2(k)=0, \text{ otherwise.} \end{cases} \tag{12}$$

where a_i is the reference acceleration at time i, $|a_{max}|$ is the maximum acceleration of the reference trajectory, and n_2 is the window size of motor-starting interval.

According to the above analysis, we select linear features $\dot{x}(k+i), i=-6,-5\ldots 6$ and nonlinear features $\{ch_1, ch_2\}$ for training of the LSTM-NN.

B. Training of LSTM-NN

Recurrent neural network (RNN) has excellent performance in model fitting of sequence-to-sequence responses but suffers from long-dependency problems. To address this issue, the LSTM-NN was proposed in [26], as a special RNN that accommodates both long-term and short-term memories by adding sequences of long-term memories to the standard RNN.

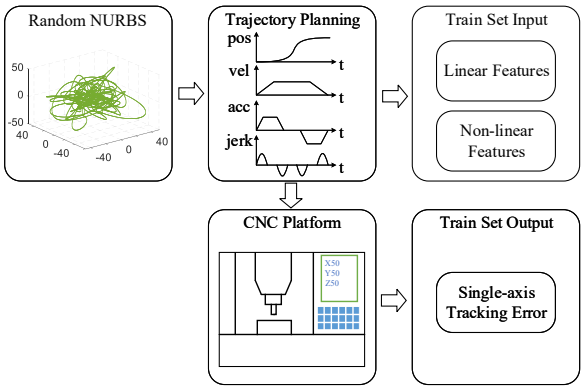


Fig. 4: Training set generation

The training set preparation is summarized in Fig. 4. First, we generate random NURBS geometric paths with random control points [22], which are composed of the independent variable vector $\vec{t}$ and the dependent variable vector $\vec{y}$, described as follows:

$$\begin{cases} \vec{t}=[t_0, t_0+\triangle t_1, \cdots, t_0+\sum_{i=0}^{n}\triangle t_i] \\ \vec{y}\sim N(\mu,\sigma^2) \end{cases} \tag{13}$$

where $\triangle t_i = 0.04 + rand(0.04, 0.08)$, and $rand(0.04, 0.08)$ represents a random number between $0.04 \sim 0.08$, $\mu = 0$ and $\sigma = 18$.

Five NURBS spline interpolations are performed on the control point sequence $f(\vec{y}, \vec{t})$, followed by $500Hz$ sampling to obtain the reference trajectory, which is used to calculate the training inputs. Simultaneously, the reference trajectory is also sent to the CNC system so we can get the actual positions. These reference positions and actual positions are used to calculate the tracking error, which will be considered as the training outputs. Finally we get the data set contains 30,106 data points, of which the first 75% for the training set and 25% for the validation set.

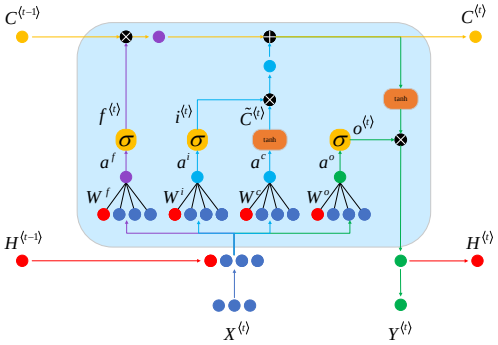


Fig. 5: Weight update

As shown in Fig. 5, the training of the LSTM-NN is to find W^f, W^i, W^c and W^o that minimize the loss function

$$J^{\langle t\rangle} = \frac{1}{2}\left\| Y^{\langle t\rangle} - {Y_d}^{\langle t\rangle} \right\|^2 \tag{14}$$

where $\|\bullet\|$ denotes the norm and ${Y_d}^{\langle t\rangle}$ is the desired output at time t. By denoting the total loss function in time duration T as $J = \sum_t^T J^{\langle t\rangle}$, the gradient of the output gate at time t is given by

$$\begin{aligned} \frac{\partial J}{\partial (W_{jk}^o)^{\langle t\rangle}} &= \frac{\partial J}{\partial o_j^{\langle t\rangle}} \frac{\partial o_j^{\langle t\rangle}}{\partial a_j^o} \frac{\partial a_j^o}{\partial (W_{jk}^o)^{\langle t\rangle}} \\ &= \frac{\partial J}{\partial H_j^{\langle t\rangle}} \odot \tanh(C_j^{\langle t\rangle}) \odot o_j^{\langle t\rangle}(1-o_j^{\langle t\rangle})(Z_k^{\langle t\rangle})^T \end{aligned} \tag{15}$$

where $(W_{jk}^o)^{\langle t\rangle}$ is the element of $(W^o)^{\langle t\rangle}$ in j-th row and k-th column.

The gradient of the input gate at time t is

$$\begin{aligned} \frac{\partial J}{\partial (W_{jk}^i)^{\langle t\rangle}} &= \frac{\partial J}{\partial C_j^{\langle t\rangle}} \frac{\partial C_j^{\langle t\rangle}}{\partial i_j^{\langle t\rangle}} \frac{\partial i_j^{\langle t\rangle}}{\partial a_j^o} \frac{\partial a_j^o}{\partial (W_{jk}^i)^{\langle t\rangle}} \\ &= \frac{\partial J}{\partial C_j^{\langle t\rangle}} \odot \tilde{C}_j^{\langle t\rangle} \odot i_j^{\langle t\rangle}(1-i_j^{\langle t\rangle})(Z_k^{\langle t\rangle})^T \end{aligned} \tag{16}$$

where $(W_{jk}^i)^{\langle t\rangle}$ is the element of $(W^i)^{\langle t\rangle}$ in j-th row and k-th column.

The gradient of cell state at time t is

$$\begin{aligned} \frac{\partial J}{\partial (W_{jk}^c)^{\langle t\rangle}} &= \frac{\partial J}{\partial C_j^{\langle t\rangle}} \frac{\partial C_j^{\langle t\rangle}}{\partial \tilde{C}_j^{\langle t\rangle}} \frac{\partial \tilde{C}_j^{\langle t\rangle}}{\partial a_j^c} \frac{\partial a_j^c}{\partial (W_{jk}^c)^{\langle t\rangle}} \\ &= \frac{\partial J}{\partial C_j^{\langle t\rangle}} \odot i_j^{\langle t\rangle} \odot (1-(\tilde{C}_j^{\langle t\rangle})^2)(Z_k^{\langle t\rangle})^T \end{aligned} \tag{17}$$

where $(W_{jk}^c)^{\langle t\rangle}$ is the element of $(W^c)^{\langle t\rangle}$ in j-th row and k-th column.

The gradient of forget gate in time t is

$$\begin{aligned}\frac{\partial J}{\partial (W_{jk}^f)^{\langle t\rangle}} &= \frac{\partial J}{\partial C_j^{\langle t\rangle}} \frac{\partial C_j^{\langle t\rangle}}{\partial \tilde{C}_j^{\langle t\rangle}} \frac{\partial \tilde{C}_j^{\langle t\rangle}}{\partial a_{jk}^c} \frac{\partial a_{jk}^c}{\partial (W_{jk}^f)^{\langle t\rangle}} \\ &= \frac{\partial J}{\partial o_j^{\langle t\rangle}} \odot C_j^{\langle t-1\rangle} \odot f_j^{\langle t\rangle}(1-f_j^{\langle t\rangle})(Z_k^{\langle t\rangle})^T\end{aligned} \tag{18}$$

where $(W_{jk}^f)^{\langle t\rangle}$ is the element of $(W^f)^{\langle t\rangle}$ in j-th row and k-th column.

In the total time duration T, we have

$$\begin{aligned}\frac{\partial J}{\partial W^o} &= \sum_t^T \frac{\partial J}{\partial (W^o)^{\langle t\rangle}}, \frac{\partial J}{\partial W^i} = \sum_t^T \frac{\partial J}{\partial (W^i)^{\langle t\rangle}} \\ \frac{\partial J}{\partial W^c} &= \sum_t^T \frac{\partial J}{\partial (W^c)^{\langle t\rangle}}, \frac{\partial J}{\partial W^f} = \sum_t^T \frac{\partial J}{\partial (W^f)^{\langle t\rangle}}\end{aligned} \tag{19}$$

Thus, the weights can be updated as

$$\begin{aligned}W^o &= W^o - \alpha \frac{\partial J}{\partial W^o}, W^i = W^i - \alpha \frac{\partial J}{\partial W^i} \\ W^c &= W^c - \alpha \frac{\partial J}{\partial W^c}, W^f = W^f - \alpha \frac{\partial J}{\partial W^f}\end{aligned} \tag{20}$$

where α is a positive learning rate.

IV. Contouring Control

With the trained LSTM-NN model of each axis in the previous section, the tracking error can be predicted, which will be further used to calculate the contour error as detailed in this section.

A. Contour error estimation and compensation

The relationship between tracking error and contour error is illustrated in Fig. 6, where P_k is the reference position, Q_k is the actual position and ε_k is the actual contour error. Suppose $\hat{e}_k^y$ and $\hat{e}_k^x$ are the predicted tracking errors on axes X and Y at instant k, then we can calculate the predicted position $\hat{Q}_k$ with the reference position, and further calculate the predicted contour error $\hat{\varepsilon}_k$ to be explained in the following.

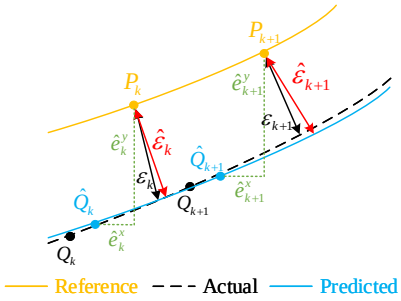


Fig. 6: The relationship between tracking error and contour error

The accuracy of contour error estimation methods in [13], [14], [15] would decrease if the sampling frequency is low, and cubic spline interpolation can effectively address this issue. Therefore, a spline-approximation-based contour error estimation method is developed to interpolate the discrete points to obtain the approximate continuous trajectory. As shown in Fig. 7, for each reference position point P_k $(k = 1, 2, 3...)$, the nearest actual position point $\{Q_w\}$ is searched among the next $\sigma + 1$ points Q_{k+i} $(i = 0, 1, 2, 3...\sigma)$, and σ varies according to reference velocity v_k and tracking error e_k:

$$\sigma = \left\lceil \frac{e_k}{v_k T} \right\rceil = \left\lceil \frac{e_k}{\left\| \overrightarrow{P_k P_{k-1}} \right\|} \right\rceil \tag{21}$$

where T is sampling period.

Then the three points $\{Q_{w-1}, Q_w, Q_{w+1}\}$ are used to perform spline interpolation, generating 10 position points. In particular, we perform Hermite cubic spline interpolation among Q_{w-1}, Q_w and Q_{w+1}, with the cubic spline expression

$$F(t) = At^3 + Bt^2 + Ct + D \tag{22}$$

where $t \in [0, 1]$. In the following, we take the cubic spline interpolation between Q_{w-1} and Q_w as an example. According to the four boundary conditions with endpoints Q_{w-1}, Q_w and endpoint tangent vectors Q'_{w-1}, Q'_w, the cubic spline expression between Q_{w-1} and Q_w can be determined:

$$\begin{aligned}F(t) = (2t^3 - 3t^2 + 1))Q_{w-1} + (-2t^3 + 3t^2))Q_w \\ + (t^3 - 2t^2 + t)Q'_{w-1} + (t^3 - t^2))Q'_w\end{aligned} \tag{23}$$

With interval of $\triangle t = 0.2$, we obtain 5 cubic spline interpolation points between Q_{w-1} and Q_w, i.e. $\{S_i \mid i \in [1, 5] \cap i \in Z^*\}$. Similarly, we obtain 5 cubic spline interpolation points between Q_w and Q_{w+1} as $\{S_i \mid i \in [6, 10] \cap i \in Z^*\}$. The shortest distance between each interpolation point and P_k is denoted as

$$\hat{\varepsilon}_k = min\left\{ \left\| \overrightarrow{P_k S_i} \right\| \right\}, i \in [1, 10] \cap i \in Z^* \tag{24}$$

The point closest to P_k is denoted as R_k, whose coordinates can be obtained.

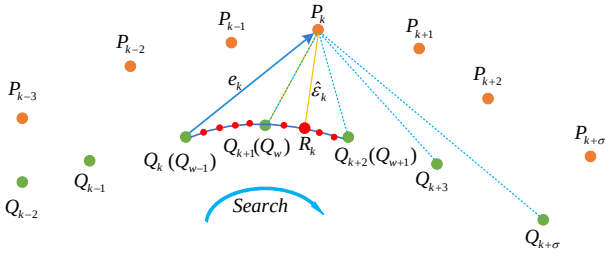


Fig. 7: Contour error estimation

With the estimated contour error, compensation at each original reference position point P_k $(k = 1, 2, 3...)$ can be performed to obtain a new reference position point P_k^{new} $(k = 1, 2, 3...)$, as illustrated in Fig. 8. In particular, the original reference position point is shifted by $\eta\hat{\varepsilon}_k$ in the direction opposite to R_k to obtain P_k^{new}, forming the reference trajectory of the CNC system after compensation.

B. Control strategy

The overall contouring control strategy is summarized in Fig. 9, which shows that the contouring control is achieved

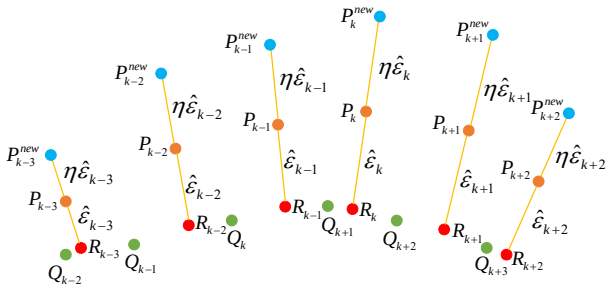


Fig. 8: Contour error compensation

by modifying the CNC system's reference trajectory generated by a trajectory planning algorithm. In particular, the control system includes two parts: controller and servo, between which there is a buffer used to store the reference position points. The method in this paper is mainly achieved by updating this buffer. First, the model of the CNC system's each axis is trained using training input and output as described in Section III-A. Then, for any given reference position point, this model will predict tracking error. The contour error estimation method in the previous subsection is used to calculate the predicted contour error, which is further used for compensation of the original reference position illustrated in Fig. 8 to obtain the new reference position. Finally, the new reference position is written into the reference position buffer to reduce the contour error.

V. Experiments

Experiments are carried out to verify the validity of the proposed contour error control method. In the experiment, we use a three-axis CNC machine tool as shown in Fig. 10, which consists of a mechanical body, three GTHD servo drives and a GSN motion control card. The sampling frequency of GSN motion control card is 500Hz. The control program is written by MFC and the controller PC has a processor of Intel i5-9400@2.9GHz.

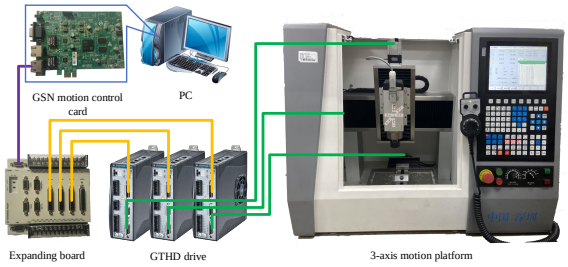


Fig. 10: Three-axis CNC machine tool based on GSN motion control card

After tuning the neural network parameters, we determine the neural network structure as shown in Figure 11, which cotains three hidden layers with 128 neurons. The solver is $Adam$, the dropout parameter is set as 0.2, the preset maximum epoch number is 1000, the minimum batch size is 128, the learning rate α is initialized to 0.01 and it is updated according to the piecewise constant attenuation. The descent factors are 125 and 0.1, respectively, the gradient threshold is 1, the verification frequency is once per 10 epochs, and the training cut-off condition is that the number of times that the verification loss is greater than or equal to the previous minimum loss is 6. The compensation gain η in Fig. 8 is set as 1.

Four typical reference curves, i.e. Crown Curve, Spiral Curve, Flower Curve and Random NURBS Curve are considered, as shown in Fig. 12. To generate reference position points, we use a trajectory planning method in [27], with the maximum velocity as $80mm/s$ and maximum acceleration as $800mm/s^2$.

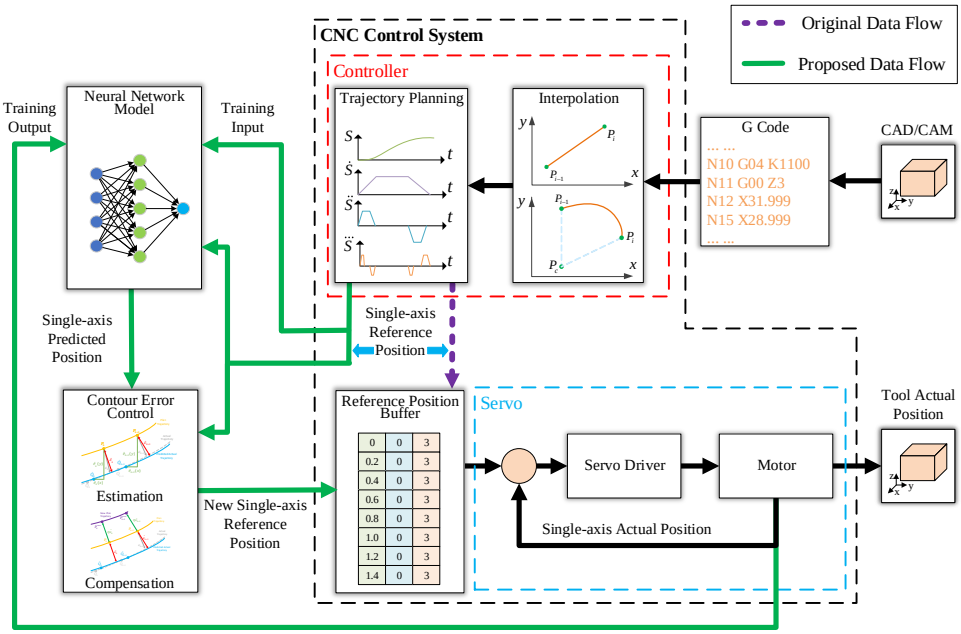


Fig. 9: The block diagram of overall control strategy

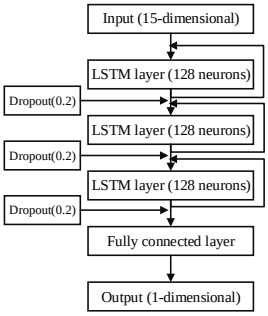
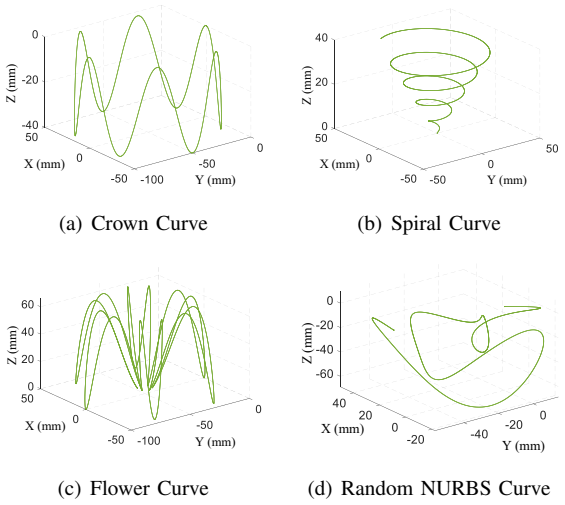


Fig. 11: The layer structure of LSTM-NN



(a) Crown Curve (b) Spiral Curve

(c) Flower Curve (d) Random NURBS Curve

Fig. 12: The reference trajectories used in experiments

A. Comparison of different contour error estimation methods

In order to demonstrate the advantages of the proposed contour error estimation method, we compare the accuracy and efficiency of different contour error estimation methods in this subsection. The test curve is a flower curve described as

$$\begin{cases} r(t) = a + bcos(ct) \\ X(t) = r(t)cos(t) \\ Y(t) = r(t)sin(t) \end{cases} \tag{25}$$

where $a = 30$, $b = 25$, $c = 6$.

The optimization method can be used to calculate the actual value of contour error if the estimated curve has a mathematical expression. In the experiment, constrained optimization function $fmincon$ is used to calculate the actual value of the contour error for comparison. We calculate the distance from actual point Q_k $(k = 1, 2, 3...)$ to the reference trajectory as the objective function, i.e.

$$\begin{aligned} f(t) &= \sqrt{|X(t) - Q_k(x)|^2 + |Y(t) - Q_k(y)|^2} \\ &= \sqrt{|r(t)cost - Q_k(x)|^2 + |r(t)sint - Q_k(y)|^2} \end{aligned} \tag{26}$$

where $Q_k(x)$ and $Q_k(y)$ represent X-axis and Y-axis position components of the actual point Q_k. In addition, the initial value t_0 of the optimization method is determined according to the parameter t corresponding to the reference point P_k, which is

$$t_0 = \begin{cases} arccos(\frac{\overrightarrow{OP_k} \cdot \overrightarrow{i}}{\|\overrightarrow{OP_k}\|\|\overrightarrow{i}\|}), \text{if } P_k(y) > 0 \\ 2\pi - arccos(\frac{\overrightarrow{OP_k} \cdot \overrightarrow{i}}{\|\overrightarrow{OP_k}\|\|\overrightarrow{i}\|}), \text{otherwise.} \end{cases} \tag{27}$$

where $\overrightarrow{i}$ is the unit vector in the positive direction of the X axis. Finally, we set constraint condition as $2\pi \geqslant t \geqslant 0$.

Table I shows that the efficiency of the proposed contour error estimation method is comparable to other methods in [13], [14], [15] and its accuracy is higher.

TABLE I: Estimation performance under different contour error estimation methods

method	max estimate error (μm)	time (s)
proposed	4.9	4.833
optimization method	-	603.7451
spline approximation [15]	14.1	3.0313
circle approximation [14]	7.5	7.1875
tangent approximation [13]	12.9	12.3281

B. Comparison of different neural network features and different neural networks

In this subsection, we compare different neural network features and different neural networks for trajectory prediction.

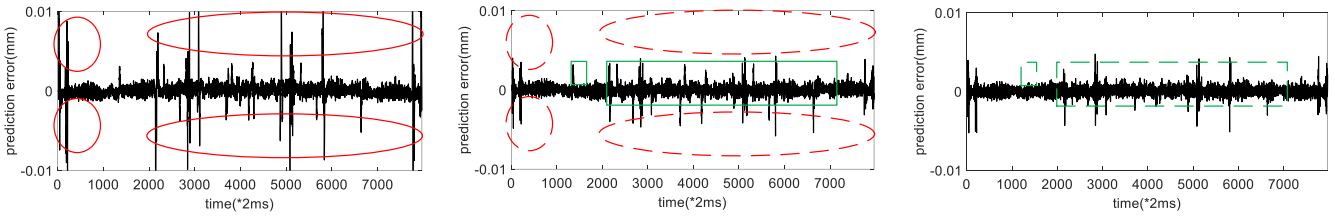
1) different neural network features: We consider the proposed feature and others in the literature, including:

i) reference velocities at the current instant and at the previous instants [28];
ii) reference position, velocity, and acceleration at the current instant [22];
iii) reference velocities at the current instant, at the previous and at the next instants, and the non-linear features (proposed).

TABLE II: Errors of X-axis tracking error predictions for a butterfly curve using different features

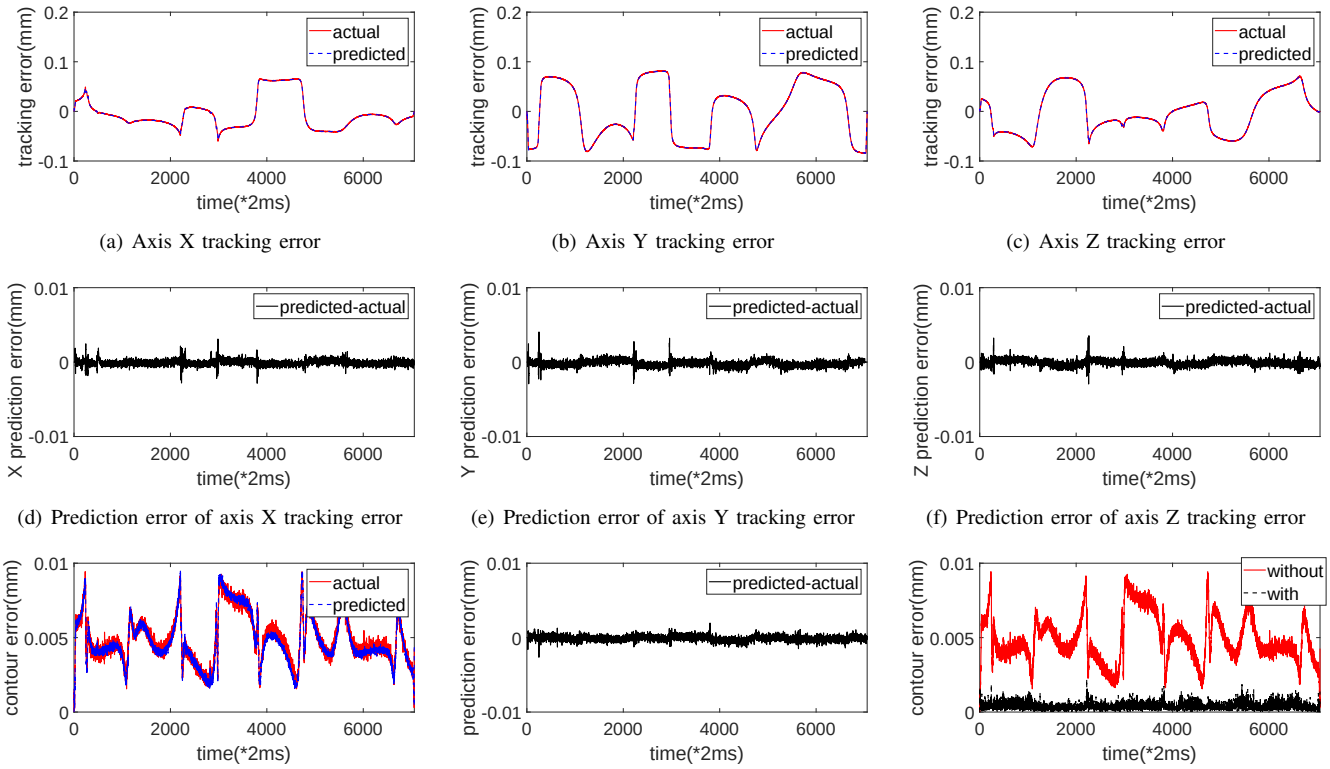
feature	MAX(μm)	RMS(μm)
i)	9.003	1.014
ii)	5.777	0.782
iii)	5.256	0.613

The comparative results for a butterfly curve are given in Fig. 13 and Table II. When feature i) is used, as the future information is not included, the red elliptical box in Fig. 13a) shows a large prediction error with the maximum (MAX) of $9.003\mu m$ and root mean square (RMS) of $1.014\mu m$. When feature ii) is used with the future information, the prediction error is slightly reduced, as shown in the green solid rectangular box in Fig. 13b), with the MAX of $5.777\mu m$ and RMS of $0.782\mu m$. When the nonlinear features are considered and the proposed feature iii) is used, the green dashed rectangular box in Fig. 13c) shows the smallest error with the MAX of $5.256\mu m$ and RMS of $0.613\mu m$.



(a) feature: $\{\dot{x}(k+i), i=-6,-5\dots 0\}$ [28] (b) feature: $\{x(k), \dot{x}(k), \ddot{x}(k)\}$ [22] (c) feature: $\{\dot{x}(k+i), i=-6,-5\dots 6\}$ & $\{ch_1, ch_2\}$

Fig. 13: Comparison of X-axis tracking error predictions for a butterfly curve using different features



(a) Axis X tracking error (b) Axis Y tracking error (c) Axis Z tracking error

(d) Prediction error of axis X tracking error (e) Prediction error of axis Y tracking error (f) Prediction error of axis Z tracking error

(g) Contour error (h) Prediction error of contour error (i) Contour error with and without compensation

Fig. 14: The prediction and compensation of the contour error for a random NURBS curve

2) different neural networks: The neural networks currently used for time series prediction include NARX, GRU, and LSTM. We perform comparison experiments using these neural networks with the same parameters. The results in Table III show that the LSTM-NN achieves the most accurate prediction.

TABLE III: Errors of X-axis tracking error predictions for a butterfly curve using different neural networks

neural network	MAX(μm)	RMS(μm)
NARX	18.52	5.262
GRU	5.811	0.709
LSTM	5.256	0.613

C. Experimental results of prediction and compensation

In this section, we show the experimental results of prediction and compensation in Fig. 14 and Tables IV & V with the following definitions. e is the tracking error, $\hat{e}$ is the predicted tracking error, and $\tilde{e}$ is the prediction error of the tracking error, i.e.

$$\tilde{e} = |e - \hat{e}| \tag{28}$$

The maximum (MAX) and root mean square (RMS) values of the tracking error are defined as

$$\tilde{e}_{max} = max\,|e-\hat{e}|\,,\ \tilde{e}_{rms} = \sqrt{\frac{1}{n}\sum |e-\hat{e}|^2} \tag{29}$$

ε is the contour error, $\hat{\varepsilon}$ is the predicted contour error, and $\tilde{\varepsilon}$ is the prediction error of the contour error, i.e.

$$\tilde{\varepsilon} = |\varepsilon - \hat{\varepsilon}| \tag{30}$$

The MAX and RMS values of the contour error are defined as

$$\tilde{\varepsilon}_{max} = max\left|\varepsilon - \hat{\varepsilon}\right|,\ \tilde{\varepsilon}_{rms} = \sqrt{\frac{1}{n}\sum\left|\varepsilon - \hat{\varepsilon}\right|^2} \tag{31}$$

We can see from Figs. 14(a)-(c) that the MAX tracking error of each axis is more than $80\mu m$ while from Figs. 14(d)-(f) that the MAX prediction error is less than $5\mu m$, indicating that the prediction of the single-axis tracking error produces excellent results in the experiment. Similar performance is achieved for the contour error prediction, with the MAX contour error of about $9\mu m$ and the MAX prediction error of less than $3\mu m$, as shown in Figs. 14(g),(h). As a result, Fig. 14(i) shows that the contour error has been effectively reduced.

Tables IV & V also indicate good performance of the prediction and compensation of contour error. For all tested reference trajectories, the prediction errors of the tracking errors are much less than the tracking errors themselves, and it is the same case for the contour error. In terms of compensation, again taking the random NRBUS curve as an example, we find that the proposed method reduces the MAX contour error from $9.435\mu m$ to $2.209\mu m$ and the RMS contour error from $5.172\mu m$ to $0.497\mu m$. Overall, the proposed method reduces the MAX contour error by more than 70% and the RMS contour error by at least 85%.

TABLE IV: Prediction performance for different trajectories

	Crown Curve	Spiral Curve	Flower Curve	Random NURBS Curve
$\tilde{e}_{max,x}(\mu m)$	4.668	3.264	4.424	3.104
$\tilde{e}_{rms,x}(\mu m)$	0.623	0.398	0.504	0.396
$\tilde{e}_{max,y}(\mu m)$	4.729	3.071	4.346	4.051
$\tilde{e}_{rms,y}(\mu m)$	0.788	0.450	0.617	0.479
$\tilde{e}_{max,z}(\mu m)$	5.262	2.958	4.905	3.543
$\tilde{e}_{rms,z}(\mu m)$	0.963	0.455	0.721	0.434
$\tilde{\varepsilon}_{max}(\mu m)$	3.553	1.982	3.644	2.640
$\tilde{\varepsilon}_{rms}(\mu m)$	0.651	0.388	0.524	0.365

TABLE V: Compensation performance for different trajectories with and without compensation

		Crown Curve	Spiral Curve	Flower Curve	Random NURBS Curve
without	$\varepsilon_{max}(\mu m)$	11.894	9.734	11.135	9.435
	$\varepsilon_{rms}(\mu m)$	7.361	5.994	5.401	5.172
with	$\varepsilon_{max}(\mu m)$	2.354	2.004	2.919	2.209
	$\varepsilon_{rms}(\mu m)$	0.527	0.399	0.544	0.497

D. Comparison of different CEC methods

At last, we perform a comparison with the ILC contour control, which is a popular method for CEC. The results in Fig. 15 and Table VI show that similar performance is achieved by ILC and the proposed method. However, it is noticed that the ILC method requires several iterations to achieve good control, and it needs to repeat the learning process when the machining trajectory changes. In comparison, the proposed method can be used for new trajectories after offline training.

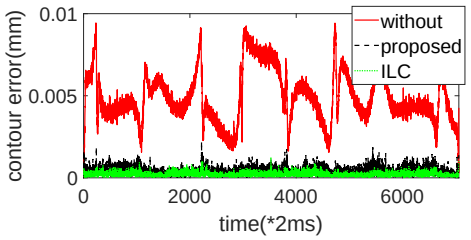


Fig. 15: The compensation of different CEC methods for a Random NURBS Curve

TABLE VI: Compensation performance of different CEC methods

CEC	$\varepsilon_{max}(\mu m)$	$\varepsilon_{rms}(\mu m)$
without	9.435	5.172
proposed	2.209	0.497
ILC	1.701	0.384

VI. CONCLUSION

This paper proposes a method for contour error prediction and compensation based on the LSTM-NN. This method is implemented by predicting the tracking error of each axis, then estimating the contour error, and finally performing compensation on the reference trajectory. On the one hand, both linear and nonlinear features are considered to improve the NN prediction performance. On the other hand, a novel contour error estimation method is developed to improve estimation performance for complex low-sampling-frequency trajectories. Comparative experiments have been carried out on typical testing trajectories to demonstrate the effectiveness and advantages of the proposed method.

REFERENCES

[1] D. Zhou, J. Gui, J. Yang, S. Li, and W. Cao, "Sliding mode control with extended state observer for multi-axis motion control system," in *2019 Chinese Control Conference (CCC)*, pp. 3196–3201, 2019.

[2] Q. Xu, "Precision motion control of piezoelectric nanopositioning stage with chattering-free adaptive sliding mode control," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 238–248, 2017.

[3] R. Ramesh, M. Mannan, and A. Poo, "Tracking and contour error control in cnc servo systems," *International Journal of Machine Tools and Manufacture*, vol. 45, no. 3, pp. 301–326, 2005.

[4] M. Tomizuka, "Zero Phase Error Tracking Algorithm for Digital Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, pp. 65–68, 03 1987.

[5] T.-C. Tsao and M. Tomizuka, "Adaptive Zero Phase Error Tracking Algorithm for Digital Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, pp. 349–354, 12 1987.

[6] S. Chen and C. Chou, "Contouring control of multi-axis motion systems for nurbs paths," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 1062–1071, 2016.

[7] R. Shi, Y. Lou, X. Zhang, and J. Li, "A novel task coordinate frame reduced- dimension 3-d contouring control," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1852–1863, 2018.

[8] Y. Koren, "Cross-Coupled Biaxial Computer Control for Manufacturing Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 265–272, 12 1980.

[9] Y. Koren, C. C. Lo, *et al.*, "Variable-gain cross-coupling controller for contouring," *Annals of the CIRP*, vol. 40, no. 1, pp. 371–374, 1991.

[10] A.-N. Poo, J. G. Bollinger, and G. W. Younkin, "Dynamic errors in type 1 contouring systems," *IEEE Transactions on Industry Applications*, no. 4, pp. 477–484, 1972.

[11] G.-C. Chiu and M. Tomizuka, "Contouring control of machine tool feed drive systems: a task coordinate frame approach," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 1, pp. 130–139, 2001.

[12] C. Hu, B. Yao, and Q. Wang, "Coordinated adaptive robust contouring control of an industrial biaxial precision gantry with cogging force compensations," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1746–1754, 2009.

[13] J. Yang and Z. Li, "A novel contour error estimation for position loop-based cross-coupled control," *IEEE/ASME transactions on mechatronics*, vol. 16, no. 4, pp. 643–655, 2010.

[14] Y.-T. Shih, C.-S. Chen, and A.-C. Lee, "A novel cross-coupling control design for bi-axis motion," *International Journal of Machine Tools and Manufacture*, vol. 42, no. 14, pp. 1539 – 1548, 2002.

[15] J. Li, Y. Wang, Y. Li, and W. Luo, "Reference trajectory modification based on spatial iterative learning for contour control of two-axis nc systems," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1266–1275, 2020.

[16] C. C. LO, "Cross-coupling control of multi-axis manufacturing systems," *Ph. D. Thesis, University of Michigan*, 1992.

[17] K. R. Simba, N. Uchiyama, and S. Sano, "Iterative contouring controller design for biaxial feed drive systems," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pp. 1–5, 2015.

[18] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5183–5189, IEEE, 2017.

[19] S. Chen and J. T. Wen, "Industrial Robot Trajectory Tracking Using Multi-Layer Neural Networks Trained by Iterative Learning Control," *arXiv e-prints*, p. arXiv:1903.00082, Feb. 2019.

[20] F. Huo and A.-N. Poo, "Nonlinear autoregressive network with exogenous inputs based contour error reduction in cnc machines," *International Journal of Machine Tools and Manufacture*, vol. 67, pp. 45–52, 2013.

[21] J. Asensio, W. Chen, and M. Tomizuka, "Feedforward input generation based on neural network prediction in multi-joint robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, no. 3, 2014.

[22] C. Hu, T. Ou, H. Chang, Y. Zhu, and L. Zhu, "Deep gru neural network prediction and feedforward compensation for precision multiaxis motion control systems," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1377–1388, 2020.

[23] G. Shen, D. Xu, L. Cao, and X. Zhu, "An improved control strategy for grid-connected voltage source inverters with an lcl filter," *IEEE Transactions on Power Electronics*, vol. 23, no. 4, pp. 1899–1906, 2008.

[24] Y. Tarng, J. Kao, and Y. Lin, "Identification of and compensation for backlash on the contouring accuracy of cnc machining centres," *The International Journal of Advanced Manufacturing Technology*, vol. 13, no. 2, pp. 77–85, 1997.

[25] S. Shi, J. Lin, X. Wang, and X. Xu, "Analysis of the transient backlash error in cnc machine tools with closed loops," *International Journal of Machine Tools and Manufacture*, vol. 93, pp. 49–60, 2015.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] Y. Chen, X. Ji, Y. Tao, and H. Wei, "Look-ahead algorithm with whole s-curve acceleration and deceleration," *Advances in Mechanical Engineering*, vol. 5, p. 974152, 2013.

[28] K. Erwinski, M. Paprocki, A. Wawrzak, and L. M. Grzesiak, "Neural network contour error predictor in cnc control systems," in *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 537–542, IEEE, 2016.

Jiangang Li (M'09-SM'20) received the B.Eng., M.Eng., Ph.D. degrees from the Xi'an Jiaotong University, China, in 1999, 2002 and 2005, respectively. From 2007 to present, he has been an Associate Professor in control science and engineering with the School of Mechanical Engineering and Automation, Harbin Institute of Technology Shenzhen, China. From 2015 to 2016, he has been a Visiting Associate in computing and mathematical sciences with the California Institute of Technology. His general research interests include high velocity and high performance control system design, motion control and motion planning.

Changgui Qi received the B.Eng. degree from the Huazhong Agricultural University, Wuhan, China, in 2018, and the M.Eng. degree from the Harbin Institute of Technology Shenzhen, China, in 2021. His research interests include motion control and machine learning.

Yanan Li (S'10-M'14-SM'21) received the B.Eng. and M.Eng. degrees from the Harbin Institute of Technology, China, in 2006 and 2008, respectively, and the Ph.D. degree from the National University of Singapore, in 2013. Currently he is a Lecturer in Control Engineering with the Department of Engineering and Design, University of Sussex, UK. From 2015 to 2017, he has been a Research Associate with the Department of Bioengineering, Imperial College London, UK. From 2013 to 2015, he has been a Research Scientist with the Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. His general research interests include human-robot interaction, robot control and control theory and applications.

Zenghao Wu received the B.Eng. degree from the Shandong Agricultural University, Shandong, China, in 2018. He is currently working towards the M.Eng. degree in the Harbin Institute of Technology Shenzhen, China. His research interests include robot control and motion control.